Neural Abstractive Summarization on Gigaword

Matthew D. Kim
Stanford University
`mdkim@stanford.edu`

Chenduo Huang
Stanford University
`cdhuang@stanford.edu`

Abstract

Text summarization has been a long-standing task. Traditional extractive summarization has its limits and the emerging abstractive summarization has been proven promising yet challenging. In this work, we explore abstractive text summarization using attentive LSTM encoder-decoder recurrent neural network to achieve sensible results. We look at potential tweaks to the conventional architecture and compare the results. Our primary dataset is the news collection by the Associated Press Worldwide from English Gigaword, a dataset of historical news articles and headlines from five reputable, international news agencies. We evaluate the effectiveness of self-trained embeddings, reversing the input text, and using a bidirectional architecture.

1 Introduction

Text summarization is the task of creating a meaningful summary for a larger piece of text. As massive amounts of data get uploaded and shared on the Internet everyday, it is more and more transparent that text summary can be utilized to help people filter and absorb information in a more efficient manner. Earlier efforts exploring this task have generally adopted an extractive approach. Summarization is achieved by choosing words from the source text while stitching them into a grammatically correct sentence. Although it is intuitive to include relevant words from source text in a summary, the extractive method have been shown to have some major limitations. Especially when the source text is long and complex, models tend to struggle because they not have the ability to paraphrase, let alone sometimes it is too hard to find where to extract the words.

From another angle, summarization can be seen as converting the input source text to the target summary text. In recent years, sequence-to-sequence models became more and more popular in the field of machine translation [Bahdanau et al., 2014] and speech recognition [Bahdanau et al., 2015]. Among them, the sequence-to-sequence model with attention are used for the summarization task [Rush et al., 2015]. Compared to previous models, this approach is fundamentally different in that it is abstractive by design. The output summary may or may not contain words from the input text and the potential of vocabulary is unlimited in a sense.

However, we cannot plainly transfer sequence-to-sequence models that do well in machine translation directly into text summarization. For one, in machine translation the output length scales with the input length, while in text summarization the summary is generally short and does not really depend on the source text. In addition, machine translation is a lossless process where the translation is expected to keep all the information in the input, while in text summarization the process is lossy by definition – we are always trying to compress and synthesize information from source text.

Our goal is to explore the relatively new and challenging field of abstractive text summarization and train a sensible model that generates summaries of new articles. We employ a LSTM encoder-decoder RNN attention model and tweaks several features of the network to compare the results.

2 Background//Related Work

2.1 Dataset

The English Gigaword Fifth Edition is approximately 27 GB of gzipped SGML-formatted newswire text data, one directory for each of the news sources. The English Gigaword was acquired over the years by LDC in the University of Pennsylvania. We decided to use the news collection by the Associated Press Worldwide, one of the 7 sources, as our dataset. We are using the first paragraph as our input source text and the headline as the gold standard output summary. Intuitively, this is a good strategy for this particular dataset. Journalists are trained to deliver the most important piece of information in the beginning of an article and the headlines are generally inferred from the same region. In the foundational work on abstractive text summary in 2016, Rush et al. also employs the same strategy, except on the entire Gigaword dataset.

Below is an example source-headline pair:

- **Source:** The British economy is poised for strong growth into 2005, raising the possibility of an interest rate hike early in the new year, according to a study published Monday.
- **Headline:** British economy poised for strong growth and possibly an interest rate hike

We can see that the first sentence contains all the words needed to construct the gold standard summary almost in the the exact form.

We also considered using the first two sentence of each article as the source text because we realized that sometimes the the second sentence can be supplementary and have some overlap in vocabulary with the headline. But in the end due to concerns of extra computations needed for each epoch, we decided against this decision.

2.2 Evolution of Neural Encoder-Decoder Models

The neural encoder-decoder models are widely used in NLP tasks involving sequence to sequence processes, such as machine text summarization [Chopra et al., 2016] and question answering [Hermann et al., 2015]. These models employ recurrent neural networks, usually LSTMs, as both their encoder and decoder. The encoder converts an input sequence into a fixed vector. The decoder then uses the vector and output a sequence of text. From a time step point of view, the encode first receives an embedded input token, which contains certain semantic meaning of the token. The hidden state of the encoder is then updated, saved, and used for calculation of the next hidden state. Once the input sequence is exhausted, the decoder is activated and calculates its initial state, usually using the last hidden state of the encoder. In a similar fashion, the decoder takes in the <GO> token, produces the next token, and uses the token to produce the next one down, cycling through the generation process until stopped.

This sequence to sequence architecture is further improved by the introduction of attention models [Bahdanau et al., 2014]. Models with attention mechanisms are more robust and versatile since it can allow decoder to focus on certain parts of the encoded input sequence at each time step of producing the next word in the output sequence.

Pointer models are another addition to the general encoder-decoder architecture [Gülçehre et al., 2016]. They are different from attention models in that they use a fussy softmax to allow an additional extractive functionality. The model essentially will evaluate to either copy a word from source text or generate a word from vocabulary. This new duality in the pointer models relieves some of the common problems in text summary. The appearance of <UNK> tokens in the summary is reduced since the input and output vocabulary is no longer fixed and the model have a more variety of words to work with. On the other hand, the model suffer less from factual errors that can easily be corrected by using some words from source text [See et al., 2017].

3 Approach

3.1 Preprocessing

Each compressed file of the Gigaword dataset contains around 10,000 articles in SGML format with sections including document metadata, headline, and paragraphs of news story text. Our first preprocessing step was to read in all of the uncompressed files and strip each article of everything except the headline (`<HEADLINE>`) and first paragraph of the news story (`<TEXT>`). Given the problem formulation, limiting the input to the first paragraph is effective because journalists and editors deliberately communicate the main idea of an article in the first paragraph.

Example Raw Gigaword Input

```
<DOC type="story" >
 <HEADLINE>
  Security Council agrees on draft resolution urging Yemen ceasefire
 </HEADLINE>
 ...
 <TEXT>
  <P>
  The Security Council agreed late Tuesday on
  the text of a draft resolution calling for an immediate   ceasefire in Yemen,
  and for the resumption of talks between northern and   southern forces, the
  council president said.
  </P>
 ...
 </TEXT>
</DOC>
```

Data cleaning involved tossing out articles with first paragraphs less than five words such as "Washington Bureau." Further cleaning involved throwing out ill-suited headlines that are either missing or all a single word in all-capitalization. We then populate two lists, one for the headlines and one for the first paragraph. In total, we had 1.3 million article-headline pairs.

3.2 Specifications

Dataset

1.3 million article-headline pairs from source APW in the Non-Annotated English Gigaword 5th Edition. Train/Evaluation/Test split was as follows: 70%, 20%, 10%.

Vocabulary

The vocabulary size was limited to 800K of the most frequent words in the article sentences.

Training

Our dataset was batched trained on size 128 (weights are updated after a batch of 128 inputs) with a learning rate of 0.5. The decay factor of the gradient descent optimizer was set to 0.99 and the dataset was trained for 15 epochs. Our max gradient norm (clipping) was 5.0.

Evaluation

We used BLEU and ROUGE-1, ROUGE-2, and ROUGE-L scores to evaluate overlap between a predicted headline and a gold standard headline. Although similar, the ROUGE-n precision is still different from the BLEU score in that BLEU score can incorporate statistics from many sizes of n-grams while ROUGE-n precision is limited to one value of n-gram. Another difference is that BLEU introduces a brevity penalty term. We decide to include BLEU scores in our evaluation, hoping to capture patterns that ROUGE-n scores are not able to show.

3.3 Architecture

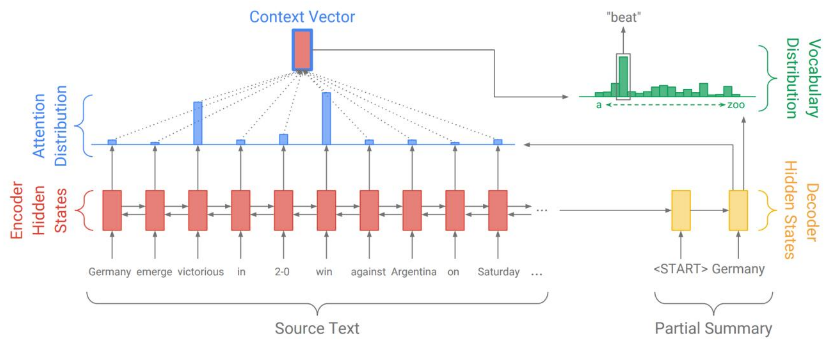


Figure 1: Discriminator architecture[1]
http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

Using TensorFlow, we modeled our first architecture from a chat bot implementation of a sequence-to-sequence encoder-decoder LSTM model[2] with attention mechanism[3]. Furthermore, we selected an output dropout rate of 0.2 between layers. In the end we choose to use three layers for our LSTM with a hidden layer size of 512. We experimented with different hyper parameters and decided this is the best architecture that achieves a balance between effectiveness and accuracy.

4 Experiments

4.1 Baseline

Our baseline is the a unidirectional LSTM-RNN encoder-decoder model with attention. During the first decode timestep, the decoder's first hidden state is calculated using the last hidden state of the encoder and a embedded <START> token that symbolizes the beginning of a sentence. For subsequent timesteps, the decoder updates its hidden state using its previous hidden state and word embedding of the previously generated token. During training, the decoder uses word embedding of the previous gold standard token instead. The generation usually stops when <END> token is generated. The baseline structurally is similar to what appeared in Bahdanau et al. [2014] except the encoder is intentionally unidirectional in this case.

4.2 Embedding

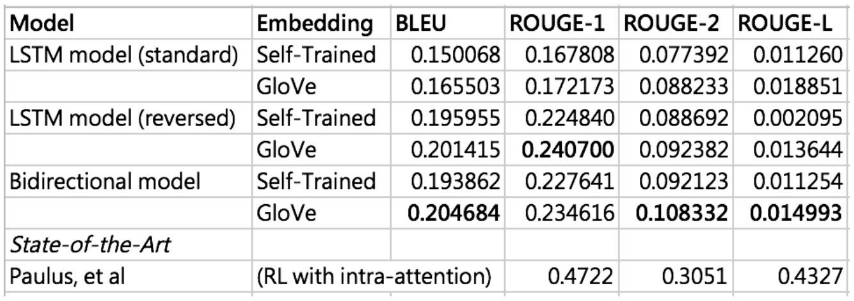
In our baseline model, we used Tensorflow to initialize the word embeddings for encoder and decoder and trained the embedding along with the other parameters of the model. The embeddings were trained against the most common 80,000 tokens in the Associated Press Worldwide dataset. We also experimented using GloVe embeddings in the model. GloVe embedding is a pre-trained on Wikipedia and Gigaword, 5th edition corpus of 6 billion tokens. This adaptation fits naturally into our task since our dataset is already a subsection of the Gigaword dataset. After employing GloVe, while we do not have a complete statistical analysis, we observe that there are less cases of <UNK> token appearing in the output, which indicates GloVe provides a good vocabulary embedding for our model.

4.3 Modifications on Baseline

Our second architectural modification is in the direction of the encoder. The baseline employs the most basic unidirection encoding. This presents a problem in that the arguably most important front part of the input text is farthest away from the decoder thus having the least impact in the hidden states of the decoder. We adopted an optimization and reversed the order of input tokens in the source text. Last but not least, inspired by [Bahdanau et al., 2014] and [See et al., 2017], we also experimented with a bidirectional LSTM-RNN encoder.

[2]https://github.com/suriyadeepan/easy_seq2seq
[3]https://github.com/tensorflow/models/tree/master/research/textsum

4.4 Results



Model	Embedding	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
LSTM model (standard)	Self-Trained	0.150068	0.167808	0.077392	0.011260
	GloVe	0.165503	0.172173	0.088233	0.018851
LSTM model (reversed)	Self-Trained	0.195955	0.224840	0.088692	0.002095
	GloVe	0.201415	**0.240700**	0.092382	0.013644
Bidirectional model	Self-Trained	0.193862	0.227641	0.092123	0.011254
	GloVe	**0.204684**	0.234616	**0.108332**	**0.014993**
State-of-the-Art					
Paulus, et al	(RL with intra-attention)		0.4722	0.3051	0.4327

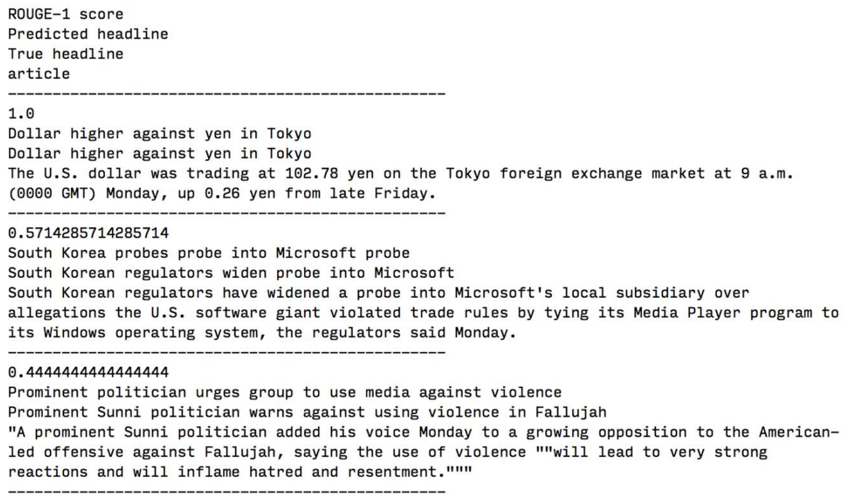
Figure 2: Test Scores for Models, Embeddings

We can confirm that the GloVe pre-trained word embeddings were more effective than our self-trained embeddings. GloVe is trained on a far larger corpus including the Gigaword dataset than our embeddings that are based only on the Associated Press Worldwide.

ROUGE-2 scores are normally lower than ROUGE-1 scores because the task of matching bigrams is more difficult. However, our ROUGE-L (longest common subsequence) scores were significantly lower than both ROUGE-2 and ROUGE-1 scores, which most likely belies an error in our code.

Our best model was the Bidirectional model on GloVe embeddings. Since hidden states were 512 units both ways, our architecture was designed to be more powerful than the other unidirectional models. However, training time also grew because of the increased number of parameters, so we had to stop training before the end of 15 epochs.

An interesting observation is that reversing the input showed better results for BLEU and ROUGE-1 in the unidirectional LSTM models. Essentially, reversing the order in the source but not the target sentence introduces short term dependencies that work in the favor of this exact problem of headline regeneration. Journalists create most headlines from the earliest and most salient portions of the first sentence of the source, so reversing the input seems to have a slight beneficial effect.



```
ROUGE-1 score
Predicted headline
True headline
article
-------------------------------------------------
1.0
Dollar higher against yen in Tokyo
Dollar higher against yen in Tokyo
The U.S. dollar was trading at 102.78 yen on the Tokyo foreign exchange market at 9 a.m.
(0000 GMT) Monday, up 0.26 yen from late Friday.
-------------------------------------------------
0.5714285714285714
South Korea probes probe into Microsoft probe
South Korean regulators widen probe into Microsoft
South Korean regulators have widened a probe into Microsoft's local subsidiary over
allegations the U.S. software giant violated trade rules by tying its Media Player program to
its Windows operating system, the regulators said Monday.
-------------------------------------------------
0.4444444444444444
Prominent politician urges group to use media against violence
Prominent Sunni politician warns against using violence in Fallujah
"A prominent Sunni politician added his voice Monday to a growing opposition to the American-
led offensive against Fallujah, saying the use of violence ""will lead to very strong
reactions and will inflame hatred and resentment."""
-------------------------------------------------
```

Figure 3: Sample output of evaluation scores in LSTM (reversed input) model, with Self-Trained embeddings

The evaluation results are generally as expected. The first sample output in Figure 3 shows a summary the model reproduced correctly. We notice that these kind of summary are generally short and succinct, and the leading paragraph generally have a distinct style, in this case, describing the stock exchange in a predictable manner. From the second input we see that our model does indeed suffer from the repetition problem. If we were to employ a better mechanism that reduces the impact of this issue, our performance will in generally become better. One hack we thought about was to do post processing on the decoder output and eliminate words that are repeating the previous word. Intuitively, this is like saying the decoder was stuttering when outputting the summary. We will manually correct the stutter. But notice that the "probe" after "Microsoft" would not go away by this post-processing. We thus discuss a more robust way to deal with this issue in the conclusion. The last example shows a summary with comparable quality to the previous example. We think this shed light to the fact that ROUGE and BLEU scores are only our best effort to quantify the result so far. We think it is also important to come up with more innovative ways to evaluate a subjective task such as text summarization.

5 Conclusion

In this paper, we introduced a combination of embedding and directional changes to an LSTM-RNN encoder-decoder model. We implemented and tested all six combinations and determined that overall the model with GloVe embedding and bidirectional encoder performed the best, but by only a small margin. We the limited size of dataset and training time held back the GloVe embedding and bidirectional encoder to become even more successful.

Future Work

While we accomplished some of our goals in this project, we are aware that we can improve our experiments in many more aspects. Above all, we would love to train and test on a bigger dataset, ideally the Gigaword dataset as a whole. We believe this will make our model more capable of summarizing news articles from all kinds of sources. In addition, we want to see if entity tagging on the input data can improve the performance of our model. Another popular extension that we would like to experiment with is the pointer model mentioned in section 2.2. We noticed that our outputs suffer from repetition of tokens. Instead of using a more brute force post-processing method, we hope that with the pointer model we will be able to achieve a same or even better effect. Last but not least, we recognize we can also employ an intra-decoder attention model to solve the repeating token problem.

Acknowledgments

We would like to give our thanks to Richard for being an amazing lecturer, Abigail for giving us patient guidance, and the rest of the CS224n staff for providing a great learning experience.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. *CoRR*, abs/1508.04395, 2015. URL `http://arxiv.org/abs/1508.04395`.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*, 2016.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *CoRR*, abs/1603.08148, 2016. URL `http://arxiv.org/abs/1603.08148`.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf`.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015. URL `http://arxiv.org/abs/1509.00685`.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017. URL `http://arxiv.org/abs/1704.04368`.